ATHiCC: An Anonymous, Asynchronous, Serverless Instant Messaging Protocol

Daniel F. Balchasan
Aalborg University
dbalchasan@gmail.com

Michal Ozaniak
Aalborg University
michal@ozaniak.sk

Yoav Schwartz
Aalborg University
yosc@itu.dk

Nicolai S. Steffensen
Aalborg University
nsst@itu.dk

Samant Khajuria
Aalborg University
skh@cmi.aau.dk

Lene T. Sørensen
Aalborg University
ls@cmi.aau.dk

Abstract

Instant messaging has become a main form of communication between people. The ability to instantly send messages to each other, even when the recipient is offline, has become second nature and is taken for granted in modern society. However, this is not without a cost. In the case of instant messaging, that cost is privacy. Service providers use centralized servers to store these messages and can collect information using the messages 'Metadata' or even read the contents of messages. This paper presents a novel protocol, ATHiCC (Asynchronous Tor Hidden Chat Communication) [1] *that allows private and anonymous communication that doesn't require a server, and yet is still able to support asynchronous communication. A simulator was implemented to test the protocol and performance under various network conditions and topologies. The results of the simulation predict high delivery rates and low delays in message delivery under most conditions, even in small network topologies.*

1. Introduction

Privacy is a fundamental human right, as recognized by the UN Declaration of Human Rights [2] and yet, recent events make it clear that internet users right to privacy is being violated. Governments are collecting more information than ever [3] and some of the biggest companies in the world use users' personal data as a financial model e.g. Facebook and Google.

Facebook, the biggest social network in the world, which was involved in a scandal leaking millions of users' details [4], has one the most popular instant messaging applications on the market [5]. In "Messenger", Facebook's instant messaging application [6], conversations are not end-to-end encrypted by default, meaning Facebook can read the majority of messages sent.

Even chat services that do implement end-to-end encryption by default can, and do, collect 'metadata'. This metadata includes, but is not limited to, the sender, receiver, and location of the message. Therefore, even though the content of the message is secure, enough information can be deduced with the metadata to undermine the user's privacy.

Chat software that do put encryption and privacy at the top of their priority, like Telegram, get attacked by governments [7], and users are getting blocked from the service. This is relatively easy to achieve through the ISPs [8], since they can be forced to block access to the servers that host the chat service.

These points have led the authors to conclude that there is a need for a fully distributed, autonomous instant messaging software that would be secure, private and anonymous, so that no user data can be collected.

There are applications [9][10] that do provide these requirements, however they are not widely used as they are less usable than their popular counterparts and users often choose what they deem is secure enough over a more secure option that is less usable [11].

One of the main features missing from these programs, is one we take for granted in modern communication systems, namely, the ability to send messages in an asynchronous manner when the recipient is offline. This is because they rely on Peer-to-Peer communication between clients which makes it challenging to support asynchronous features.

The goal of this paper is to suggest a protocol that supports asynchronous messaging while being private, secure and fully distributed. This work is done by designing a protocol that uses other users in the network, in a unique fashion, while relying on the characteristics of Tor onion services to deliver asynchronous messages. The protocol is then tested using a simulation software developed to test the protocol's performance under varying conditions.

The outline of this paper is as follows; in Section 2, an overview of related work is presented. In Section 3 the conceptual framework around the protocol is laid out. Section 4 and Section 5 give the background and

URI: https://hdl.handle.net/10125/59942
ISBN: 978-0-9981331-2-6

details of the protocol. Section 6. contains the methodology for the simulation, and in Section 7 the results of the simulation are presented. Finally, Section 8 and Section 9 discuss the considerations made in the design process and possible future work.

2. Related Work

Much work has been put into developing solutions for secure and private end-to-end encrypted communication. Many of these approaches either offer high levels of security and anonymity, but with a low number of features with respect to messaging. Others offer many features, but lower levels of security and anonymity.

2.1. Ricochet

Ricochet [10] is an example of a chat application which offers high-level security and anonymity by utilizing Tor [12] and Tor Onion Services [13]. It utilizes end-to-end encryption and guarantees that only the sender and receiver can read the content.

By utilizing Tor Onion Services, it also eliminates any possibility that an entity can gather metadata or track who send which message. Furthermore, the application works autonomously without the need of any kind of servers for routing or connecting peers, since this is done by the Tor network. This means no one can track who is using the application, as all Tor traffic is indistinguishable [14]. This level of security and anonymity does come with a downside as Ricochet only works when both parties are online at the same time.

2.2. Signal

Signal is a chat protocol, developed by Open Whispers Systems in 2013 [15] and implemented into a number of different chat applications like, Signal [16], WhatsApp [17], secret conversations in Facebook Messenger and Google Allo in incognito mode [18]. It offers a high level of security by enabling end-to-end encryption, with a different key for each message. Providing perfect forward secrecy, so that if one key is lost, no other messages can be decrypted.

In Signal all messages are sent to a server, making it possible to send messages to users who are offline. This doesn't allow the server to see the content of the messages, but all metadata can be collected. Signal is therefore considered as a secure, but not a private chat protocol.

2.3. Tox

Tox [9] is an encrypted instant messaging protocol, which provides peer-to-peer communication. It works by creating a network of users, who via an anonymous identifier connect and send messages to each other. The protocol employs perfect forward secrecy, just like Signal does.

Tox doesn't natively support asynchronous messages, it only implements a 'pseudo-offline' message [19], where a message is stored locally at the user, until both are online. Third-party developers have tried to solve this issue of Asynchronous messaging in two different ways: Relay through another user and relay through a decentralized server, called 'supernodes'.

None of the presented solutions combine both high levels of security and the possibility to send asynchronous messages and to the best of the authors knowledge, no other current solution on the market offer this. In this paper, such a solution will be presented.

3. Conceptual Framework

In this section we will present some of the concepts and terms needed in order to understand the design and functionality of the solution presented in this paper.

3.1. Tor

One of the main requirements set for the protocol is anonymity. In the context of this paper, maintaining anonymity means not disclosing any information which may indicate the identity of the user or their location, namely the IP address of the device.

It was decided that the IP address would be kept private by designing the protocol over an Onion Routing network [18].

Onion Routing [20] provides anonymous routing of data over the internet, by encapsulating data packets (including the IP layer headers) in encryption layers like an onion (hence the name). These packets are then sent through multiple proxies (Onion Routers), each removing a layer of encryption until the clear-text packet is sent by the last proxy to the destination.

The Onion Routing network selected for the protocol is Tor [12], due to the scale and performance. In order to open secure connections over the Tor network, first a list of onion routers must be retrieved from a distributed hash table. Using this list, three random onion routers are picked which will function as proxies. Shared keys are then negotiated via TLS/SSLv3 with each of the onion routers and a path (tunnel) in the network is created.

When using Tor onion routing, even though the base packets are sent over TCP/IP, they cannot be traced back to their source, even by the receiver of the packet. However, as the packets leave the last leg of the path unencrypted, the payloads of said packets are not kept secure.

3.2. Tor Onion Services

Onion services [13] is a feature of the Tor network. Previously known as 'Hidden Service', Onion Services allow devices to provide services over the Tor network, without revealing their IP addresses, and thus their location.

Onion services are made possible by having the service provider join the Tor network. This is done by the service creating a public-private key pair and an Onion address, as a subset of their public key. This Onion address is then published on the Tor distributed hash table, as well as addresses of other nodes known as 'introduction points' (IP). The IP make it possible to reach the Onion service provider, by forwarding packets through previously defined Tor connections to the service provider. Using the IP, a 'Rendezvous Point' (another Tor router) is agreed upon, and Tor connections and established to it by the requester and the service provider.

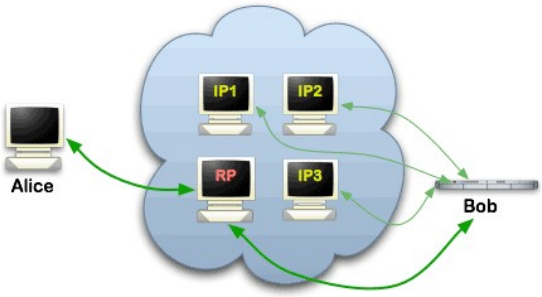

Figure 1: Communication from a client to a Tor Onion Services, by using a rendezvous.

Onion services provide multiple benefits to normal onion routing. First, they provide end-to-end address and payload encryption. Second, they don't require that the sender of a packet know the IP address of the receiver. Instead they use Onion addresses which cannot be traced back to their owner. Third, by implementing Onion services on both communicating parties, a full duplex communication can be implemented which only sends packets through the Tor network, without sharing anything more than randomly generated Onion addresses.

3.3. Centralized/decentralized/distributed

Centralized Systems are systems which rely on a single entity for decision making, such as a server. This means that one entity (or a group of entities acting as one) provides a critical service for the function of the system.

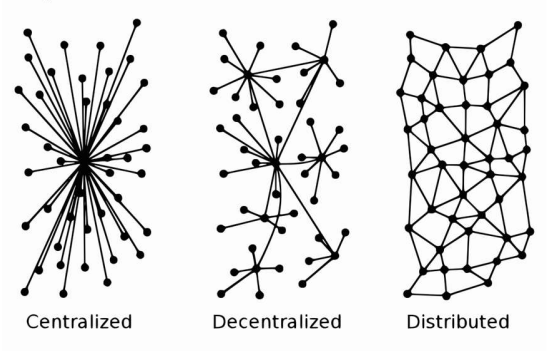

Figure 2: Difference between centralized, decentralized and distributed networks.

Centralized systems, in server-client paradigms, are very common, as they are easy to maintain, manage and provide easy control of the system.

Decentralized system are systems where a subset of the entities provides services and decision making to the rest of the network. These service-providing entities work independently of each other.

Distributed systems are systems where all entities are equal. There are no nodes who are more 'important' than other, and decision making is made on the individual level.

3.4. Synchronous/asynchronous

The basic functionality of any messaging application can be split into 2 kinds of messages. Synchronous and asynchronous messages. Synchronous communication refers to data sent while the recipient of the data is active and available. This means that in order to be able to use the service, both sender and receiver of the messages must be 'online' at the time of sending the messages.

Asynchronous messages refer to data which is sent while the recipient is not available. This data will be received by the recipient when, or shortly after, coming online. After sending the messages, 3rd parties provide services to ensure message delivery.

4. Synchronous messaging

The first, and simpler, aspect of the ATHiCC protocol relates to synchronous messages. These messages are to be sent directly, and as quickly as possible, to the other side of the conversation.

Due to the requirement of ATHiCC to be serverless, synchronous messages are sent in a peer-to-peer method.

In order to allow connections between nodes in the network, while maintaining anonymity, ATHiCC uses Tor Onion Services in order to establish the connections. In order for a node to use the protocol, they must create an Onion Service, and therefore generate an Onion Address. The Onion Address is then exchanged off-band with other users who wish to communicate over the protocol. In the asynchronous messaging section, this Onion Address is referred to as the 'private address'.

Whenever a node requests to open a session (start a conversation) with another node, the former must use the latter's Onion Address in order to contact them. Once the Tor connection is established, and a connection request is received, the receiver authenticates the requester's Onion Address. If the address is approved by the receiver, a communication channel is established.

After the first time an address is approved, subsequent connections will be accepted with an 'already known' response in the authentication process. Once 2 nodes have performed the first authentication and connection, they are considered 'contacts' of each other.

As every node has their own Onion Service, they can be contacted, and receive connection requests ('Contact' requests) from any other node who has their Onion Address. Therefore, the Onion Address should be kept private and shared only with contacts one wishes to establish connections with.

Once the 'contact' relationship is established, the nodes are able to freely send packets over a TCP connection through a secure tunnel created by the Tor network.

5. Asynchronous messaging

The main difficulty of sending Asynchronous messages in a distributed system comes from the need of 3rd party services for hosting of messages between sending and delivery, as there are no entities in the network which guarantee their availability (as is common in centralized and decentralized networks).

The solution proposed in this paper was named the 'P.O Box' solution due to its similarity to services offered by the Post Office. In this solution other nodes in the network provide message hosting services while the recipient is offline.

This section will describe the solution designed to provide the Asynchronous messaging features of ATHiCC.

5.1. Double addresses

As with traditional P.O Boxes, every node which provides messages hosting services has a unique identifier. In ATHiCC, this unique identifier is an Onion Address, used in order to access the Onion service offered by the P.O Box.

However, the Onion address of a node is also used in order to send 'Contact requests' and messages to each node. In order to provide privacy and avoid social engineering techniques, such as phishing, the ATHiCC protocol defines both a private and a public onion address for each node. The private address is shared and used for Synchronous messaging and 'Contact requests', while the public addresses are only used for providing P.O Box services.

By defining a public and private address for each node, the protocol also maintains privacy of addresses in small networks where the topology could be inferred due to the distribution of P.O Box addresses.

5.2. The P.O box solution

The P.O Box solution provides Asynchronous messaging by utilizing message hosting services provided by other nodes in the network.

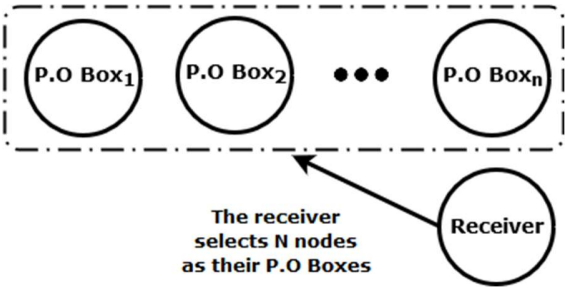

Figure 3: First step of asynchronous messaging.

The first step of the Asynchronous messaging starts with each node performing a 'P.O Box association' process, in which a node (the 'P.O Box') agrees to provide message hosting services for another node (the 'Receiver'). This association process produces a P.O Box Token, which contains needed information such as P.O Box address, association unique identifier and proof of ownership of the token.

In order to increase the reliability of the protocol, each Receiver node performs P.O Box association processes with multiple nodes, and thus has multiple P.O Boxes to use. Figure 3 depicts the first step of the asynchronous messaging process.

Once a P.O Box association process is complete, the Receiver node publishes their Token (excluding the proof of ownership) to all their contacts, informing

them of the P.O Boxes the Receiver node is using. Token publishing is performed synchronously whenever nodes connect for the first time or whenever they have a synchronous connection. Token publishing is also performed Asynchronously when a new P.O Box association is performed. Figure 4 depicts the second step.

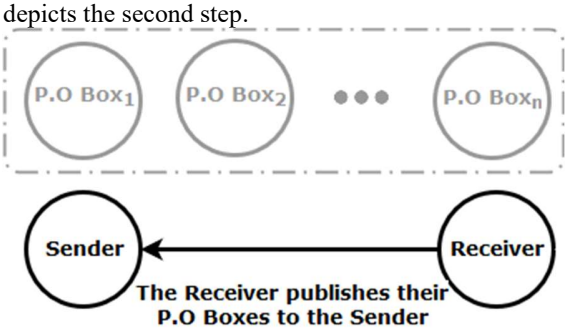

Figure 4: Second step of asynchronous messaging.

By presenting the Token to the P.O Box the Sender shows proof that they are allowed to upload Asynchronous messages to P.O Box, addressed to the owner of the token. In order to increase the reliability of the protocol, the Sender will upload the same message to multiple P.O Boxes used by the Receiver.

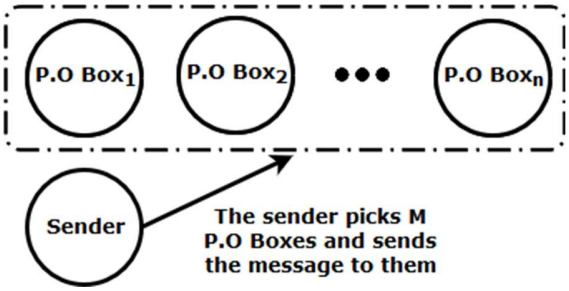

Figure 5: Third step of asynchronous messaging.

Once messages are uploaded to the P.O Box, the Receiver node is able to collect them at any time the P.O Box is online. Once the Receiver node comes online, they query all their online P.O boxes for messages, presenting the 'proof of ownership' agreed upon during the association process. Until all P.O boxes have been queried, the Receiver node would continue to query them at fixed intervals.

The protocol relies on probability. Out of the list of P.O. Boxes used by the Receiver, at least one P.O Box would need to be online both at the time of sending the message and while the Receiver is online next. By adjusting the number of P.O Boxes used by the nodes, a balance between reliability and network load can be achieved.

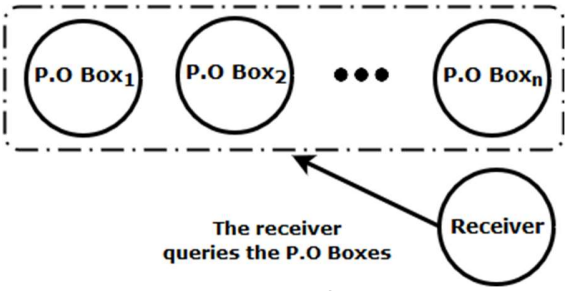

Figure 6: Fourth step of asynchronous messaging.

In order to address the potential of missed messages, the protocol includes a process for repeating previously sent messages in a sliding window, as well as a synchronous 'Synchronization' process used to exchange previously missed messages, once Sender and Receiver are online at the same time.

Finally, in order to address potential misuse of the protocol, it defines methods for ranking and 'blacklisting' nodes based on their success rates and misuse. Identification of misuse is done using confirmation receipts from the P.O Box to the Sender, which are then presented to the Receiver during the synchronous 'Synchronization' process.

5.3. Distribution

The distributed nature of the solution presents a major challenge. As there are no central entities in the network, the only nodes known to each node are those they know as 'Contacts'. Relying on contacts alone to perform P.O Box services would mean a low reliability for nodes with few contacts, as well as breach the privacy of the users by publishing a list of their contacts as P.O Boxes.

In order to allow any node to perform the role of P.O. Box for any other node, the protocol defines a propagation method for addresses. This process allows the public addresses of nodes to be published throughout the network, so they may provide services to previously unknown nodes.

To support address propagation, the protocol relies on the publishing of Tokens, used as part of the asynchronous messaging process. As each Token includes the public address of the P.O, contacts of a specific node learn the public addresses known by that node. Once they learn a new public address, they may initiate P.O Box association with that node as well, depending on their current need for more P.O Boxes. Otherwise, the contacts would add the address to their list of 'known P.O Boxes'. Figure 7 shows the propagation of addresses by Token publishing.

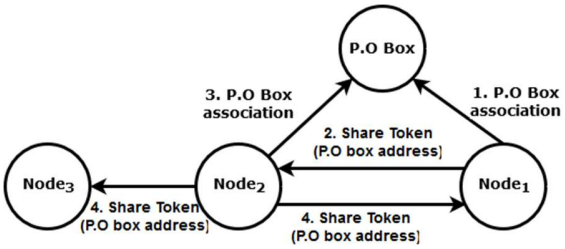

Figure 7: Propagation of P.O. Box addresses by Token publishing.

As each node in the network shares its own P.O Boxes, the addresses would propagate throughout the connected network graph. In order to facilitate the spread of addresses, and to distribute the load on the different nodes, P.O Box associations are maintained for a fixed period of time. Once an association 'times out', a new node is selected at random from the list of known public addresses, with exclusion of addresses which are 'blacklisted' or have previously shown low reliability, when compared to the other known nodes.

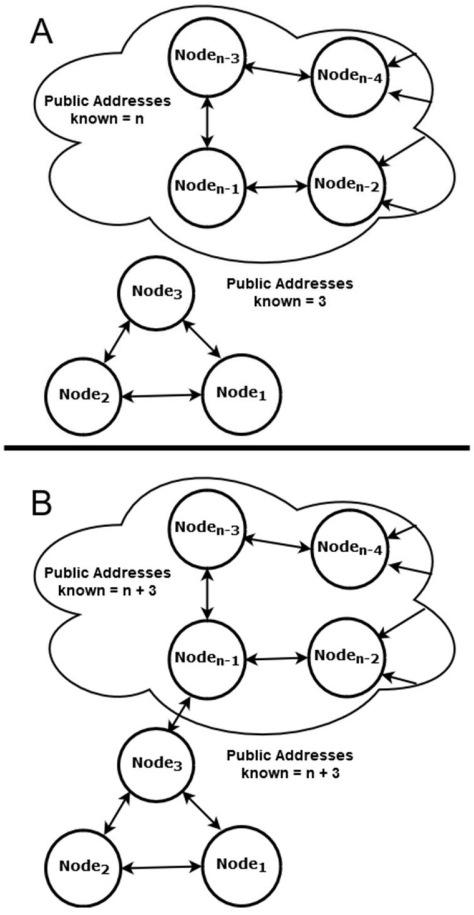

Figure 8: States of a small network before (A) and after (B) connecting to a larger network.

This method has a weakness in small, closed networks of contacts. Should a small number of nodes all share the same connections to other nodes, the only nodes available to propagate are in that subset. However, once the small network is connected to another network through any node, all connected addresses should spread. Figure 8 shows the states for a small network before and after connecting to a larger network though a single node. This behavior is simulated in the simulation when address propagation is looked into.

6. Simulation

To test that the protocol works and scales, independent of the network size, or network topology, a verification or test method is needed. We have identified two key aspects of the protocol to test which are needed to assess its viability. These are the 'messages sent and receive success rate', and the 'propagation of P.O. Box addresses'. For this purpose, a network simulation was designed and implemented [21].

The simulation was designed to test several hypotheses. These hypotheses helped scope the simulation and focused the analysis of the data, as well as assessing if the simulation worked as expected. For the analysis, we assumed that the simulation worked as expected, meaning no errors or bugs in the code were present to an extent where the data would be corrupted.

The hypotheses were split into two different parts:

Propagation of P.O. Box addresses and Asynchronous message delivery. Where only one hypothesis was tested regarding P.O. Box address propagation and several regarding Asynchronous message delivery. Below are the hypotheses that were tested:

1. The average number of known P.O. Box addresses will increase over time.
2. The more time users spend online in a given period, the higher percentage of Asynchronous messages will get delivered.
3. The more time users spend online in a given period, the faster Asynchronous messages will be delivered.
4. The more P.O. Boxes a user sends an Asynchronous message to, the higher percentage of the messages will get delivered.
5. The more P.O. Boxes a user sends an Asynchronous message to, the faster the messages will arrive.

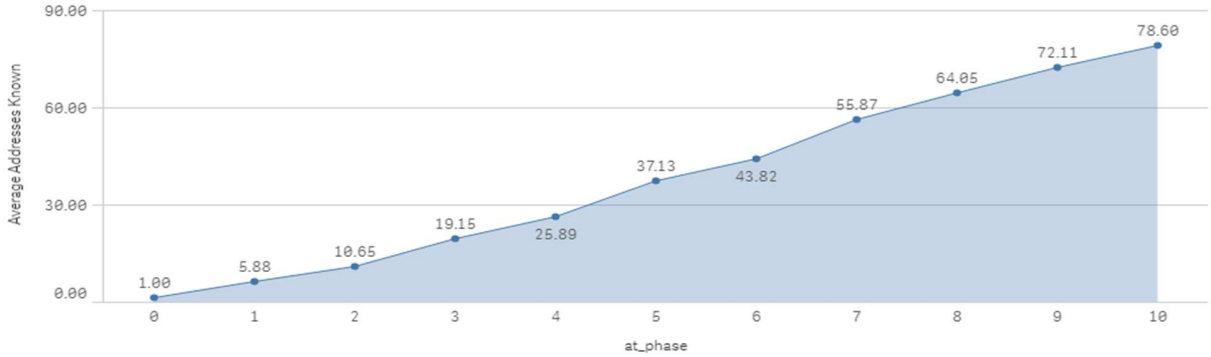

Figure 9: Average number of known P.O. Box addresses at each phase

In the simulation two types of datasets were used. First, a small star topology was generated. It was used to test that the protocol could work in small topologies, and for testing Asynchronous messaging success rates. A small topology is sufficient as the size of the network is not a significant factor for messages. This is due to the limited number of nodes involved in asynchronous message sending, in contrast to address propagation.

For large scale testing, mainly to test P.O. Box propagation, it was necessary to get a topology that is large enough and made up of different sub topologies. In order to best simulate the behavior of a system based on the THiCC protocol in a real-world environment, the network topology should resemble a real-world network as closely as possible.

Furthermore, it needed to be temporal, meaning new connections between nodes are timestamped. This is so we can observe the graph growth over time and simulate how a network would grow. Since an instant messaging is made up of friends or people who have a relationship, it would likely take a similar topology to a social network, which is a complex network [22]. There are two ways to get such a topology: generate one using an algorithm; or use an existing topology. To make sure the simulation is as close to reality as possible, a real network topology was used from the Digg social network [23]. The Digg topology is made up of millions of nodes, so a slice in time was used instead of the full topology.

In a real scenario, every node in the network is a real person with a different use case and behavior, mostly different online times and online periods. In the simulation all P.O. Boxes and clients share the same overall online time statistics. This also means that the algorithm for prioritizing P.O. Boxes was not implemented since its irrelevant to prioritize identical nodes.

6.2. Data output

The simulator takes a configuration file as an input. This configuration file includes the global variables that affect the behavior of the network or the nodes. These configurations control for example the topology to use, online frequency for the nodes and the number of P.O. Boxes to use.

For each of these configuration inputs, the simulation outputs data. First, is every message that was sent including its origin, target, sent time and optionally received time. With this data, it is possible to determine the success rate of messages sent, as well as the average time a message takes to arrive. Second, the simulation outputs which P.O. Boxes were known to which nodes and at what time-point. Using this information, it is possible to map out the rate and efficiency at which P.O. Boxes propagate through the system.

7. Results

First, we set out to test the hypothesis regarding the propagation of P.O. Box addresses throughout a large network. To test this hypothesis, a simulation was run on a large-scale network, using a topology from a social network site. The iteration when an address was shared to each node was logged and grouped into 10 equal phases (for ease of analysis).

Figure 9 shows the average number of P.O Box addresses known by the nodes in the network, over the ‘phase’. One phase is simply a tenth of the iterations. As seen in Figure 9, the results of the simulation support the first hypothesis that he ‘the average number of known P.O. Box addresses will increase over time’, by showing that average known number of P.O. Boxes increases steadily over the separate phases. This suggests that given enough time, a large network which is structured and grows like a real social

network would allow the nodes to learn new addresses.

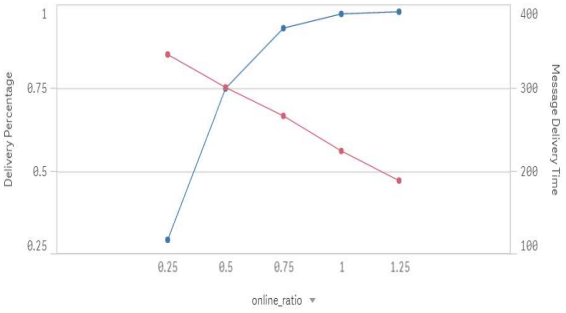

Figure 10: Delivery percentage (blue) and message delivery time (red) by differing online ratios

The second and third hypothesis refer to the online time or 'online ratio'. For this purpose, the simulation included several test runs. Each run was configured to run with identical configurations, with the only difference being the 'online ratio' (the percentage of time the node spends online in a period). The expectation is that the longer a node is online, the better chances it has to deliver a message and will therefore do so faster.

As seen in Figure 10, the delivery percentage increases and the message delivery time decreases as the online ratio increases. This means that the more time a node spends online, the more likely it is for that node to receive the asynchronous messages sent to it, and the faster it will receive them. These observations support the hypothesis presented earlier and suggests that the simulation behaves as expected.

Another consideration is what happens when a node wanted to send an asynchronous message, but it did not have any online P.O. Boxes to deliver the message to. This would not register as a message that was sent and not delivered, it would instead not be sent at all. However, for the receiving node the result is similar, where the asynchronous message did not come across. Figure 11 illustrates how many messages failed to upload to P.O Boxes at all at each different online ratio.

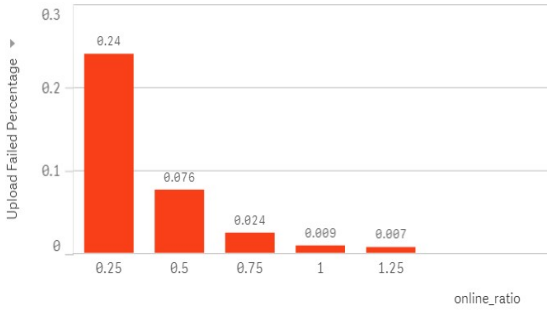

Figure 11: Upload failed percentage over different online ratios.

The results of the simulation suggest that a network with nodes which is spend less time online are less likely to be able to send the messages at all. Further analysis shows that any message which has failed to upload was attempted during the first quarter of the simulation. This suggests that this behavior is caused by a low propagation of P.O Box addresses in the early stages of the network.

These final two hypotheses consider the number of P.O. Boxes used in the delivery of messages. The expectation is that the more boxes used, the higher the chance the messages will be delivered and the less time it would take. Figure 12 shows the delivery percentage and message delivery time by the number of P.O. Boxes used by each message.

As expected, the observation shows that an increase in the number of P.O. Boxes used has an impact on both the message delivery time and delivery percentage. The higher the number of P.O. Boxes used, the more reliable and the higher performance you get from the protocol in terms of delivery percentage and message delivery time.

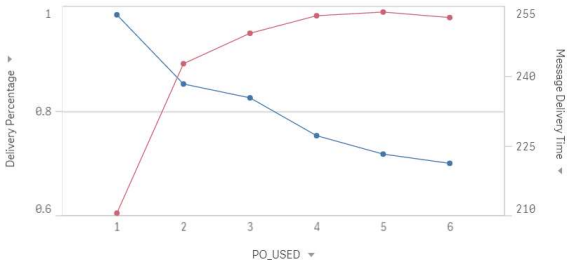

Figure 12: Delivery percentage (red) and message delivery time (blue) by different number of P.O. Boxes used.

To support this observation, a significance test was conducted which shows that the delivery percentage and message delivery time' are significantly** different from all the values being equal. Furthermore, no significant difference could be found from choosing four, five or six P.O boxes. In addition, a F-test showed significant** difference between three and four P.O Boxes used. This suggests that four P.O. Boxes is fitting number for the configuration, given the other configurations used.

The hypothesis which were presented and analyzed were helpful to determine how the protocol behaves in a simulated environment, and the review the reliability of the simulator to work as expected. In addition to reviewing the way the nodes using the protocol can send and receive messages, the overall performance of the protocol can be reviewed. As seen the in previous figures, the protocol can reach message delivery

probabilities of about 98-99% in the simulated environment, depending on the configurations used.

While this does not guarantee a reliable performance in a real-life environment, it suggests that the protocol could work and is worth further investigation.

8. Discussion

In this research paper we have presented a novel, fully distributed, private and anonymous instant messaging protocol, which also supports asynchronous messaging. This protocol is built on top of Tor onion services.

Analyzing the results of the simulation, we predict that the protocol will perform with a high degree of reliability and low delays in delivering messages. However, since the protocol doesn't use name servers due to the requirement of full distribution, different networks of users are not forced to merge, and therefore, situations may occur where a network is too small to have enough P.O. boxes to provide the redundancy needed for high degrees of reliability. However, this is a small limitation, as the protocol is designed to ensure that once any node in a network connects to a node from a different network, the P.O. box addresses will propagate.

To verify and test the protocol, we used a simulation software, intended to test the key aspects of the protocol such as the P.O. box propagation and message delivery rates and delays. However, the simulation is implemented such that all the nodes have the same Online/Offline characteristics. This means all nodes, on average, are online the same proportion of the simulation time (but not necessarily the same times). This behavior is unlikely in a "real life" environment, where users have varying behaviors and patterns, and it may be worth running the simulation with real user online time data. Yet, we predict it would not vary the results significantly.

Another consideration was whether or not to formally verify the protocol [24][25]. On the one hand, verifying the protocol could definitely prove the it would work, on the other hand, it would not indicate anything about performance, which is critical in this case, and therefore a simulation was chosen.

An implementation of ATHiCC would surely be in a context, where total anonymity and high levels of security is required. Such implementations would include messaging software for intelligence services and undercover journalist, for whom being anonymous and hidden might in extreme cases mean the difference between life and death. Military use would also be suitable case, in times when existing network infrastructure can be used.

The authors are aware that this protocol might be used by individuals with malicious intents, to communicate and plan, limiting the possibilities law enforcement agencies might have. We believe that it is impossible to create technologies which is only used for good intents and this protocol is no different.

9. Conclusion and Future Work

In this paper, a design of an instant messaging protocol is described. The aim was to allow asynchronous messaging while also maintaining privacy, anonymity, security and availability of the system for the users. After the protocol was designed, a simulation was performed, and the results show that the protocol performs as expected, even under strained conditions.

In the future, we would seek to add the ability to conduct group chats as we deem this the most important feature to allow a software based on the protocol to compete in the current market of instant messaging applications. This is however, technically challenging, as currently, asynchronous group encryption remains an unsolved problem if we exclude N times protocols where a key exchange with each person in the group is required. This is especially true without a central entity like a server. However, future work on the MLS protocol [26] may prove to be a possible solution.

10. References

[1] D. Balchasan, N. S. Steffensen, M. Ozaniak, and Y. Schwartz, "ATHiCC - Asynchronous Tor Hidden Chat Communication," Copenhagen, 2018.

[2] The United Nations, *Universal Declaration of Human Rights*. 1948.

[3] S. Landau, "Making Sense from Snowden: What's Significant in the NSA Surveillance Revelations," *IEEE Secur. Priv.*, vol. 11, no. 4, pp. 54–63, 2013.

[4] K. Granville, "Facebook and Cambridge Analytica: What You Need to Know as Fallout Widens," *The New York Times*, 2018. [Online]. Available: https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html. [Accessed: 22-May-2018].

[5] "Most popular messaging apps 2018," *Statista*, 2018. [Online]. Available: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/. [Accessed: 13-Jun-2018].

[6] "Messenger." [Online]. Available:

https://www.messenger.com/. [Accessed: 13-Jun-2018].
[7] T. Warren, “Russia orders immediate block of Telegram messaging app,” *The Verge*, 13-Apr-2018.
[8] M. Dornseif, “Government mandated blocking of foreign Web content,” *CoRR*, vol. cs.CY/0404, 2004.
[9] “Tox,” 2018. [Online]. Available: https://tox.chat/. [Accessed: 22-May-2018].
[10] “Ricochet.” [Online]. Available: https://ricochet.im/. [Accessed: 13-Jun-2018].
[11] W. Bai, M. Namara, Y. Qian, P. G. Kelley, M. L. Mazurek, and D. Kim, “An Inconvenient Trust: User Attitudes toward Security and Usability Tradeoffs for Key-Directory Encryption Systems,” in *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)*, 2016, pp. 113–130.
[12] “Tor.” [Online]. Available: https://www.torproject.org/. [Accessed: 17-Dec-2017].
[13] “Tor - Onion Service.” [Online]. Available: https://www.torproject.org/docs/onion-services.html.en. [Accessed: 17-Dec-2017].
[14] G. Owen and N. Savage, “Empirical analysis of Tor Hidden Services,” *IET Inf. Secur.*, vol. 10, no. 3, pp. 113–118, 2016.
[15] “Signal Protocol.” [Online]. Available: https://signal.org/blog/advanced-ratcheting/. [Accessed: 17-Dec-2017].
[16] “Signal.” [Online]. Available: https://signal.org/. [Accessed: 17-Dec-2017].
[17] “WhatsApp.” [Online]. Available: https://www.whatsapp.com/. [Accessed: 17-Dec-2017].
[18] “Google Allo - A smart messaging app.” [Online]. Available: https://allo.google.com/. [Accessed: 13-Jun-2018].
[19] “Tox Wiki - Offline Messaging,” 2018. [Online]. Available: https://wiki.tox.chat/users/offline_messaging. [Accessed: 22-May-2018].
[20] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, “Anonymous Connections and Onion Routing,” *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, p. 15, 1998.
[21] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, and G. F. Riley, “Large-scale network simulation: how big? how fast?,” in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, 2003, pp. 116–123.
[22] S. H. Strogatz, “Exploring Complex Networks,” *Nature*, vol. 410, no. 6825, pp. 268–276, Mar. 2001.
[23] “Digg,” 2018. [Online]. Available: http://digg.com/. [Accessed: 22-May-2018].
[24] R. Lai and A. Jirachiefpattana, “Protocol Verification,” in *Communication Protocol Specification and Verification*, Boston, MA: Springer US, 1998, pp. 143–163.
[25] M. G. Gouda, “Protocol verification made simple: a tutorial,” *Comput. Networks ISDN Syst.*, vol. 25, no. 9, pp. 969–980, 1993.
[26] R. Barnes, J. Millican, E. Omara, K. Cohn-Gordon, and Robert. R, “The Messaging Layer Security (MLS) Protocol,” 2018.